Input/Output in C

1. Input/Output a caratteri:

int **getchar**(void); legge un carattere
- restituisce il carattere letto **convertito in int** o **EOF** in caso di end-of-file o errore

int **putchar**(int c); scrive un carattere
- restituisce il carattere scritto o **EOF** in caso di errore

Esempio: Programma che copia da input (la tastiera) su output (il video):

```
#include <stdio.h>
main()
    {
    int c;
    while ((c = getchar()) != EOF)  putchar(c);
    }
```

ATTENZIONE: La funzione getchar comincia a restituire caratteri solo quando è stato battuto un carriage return (invio) e il sistema operativo li ha memorizzati

2. Input/Output a stringhe di caratteri:

char ***gets**(char *s); legge una stringa
- restituisce la stringa
  se ok, indirizzo del primo carattere o
  in caso di end-of-file o errore
  stringa nulla (**0** ossia carattere NULL)

int **puts**(char *s); scrive una stringa
- in caso di errore restituisce **EOF**

Esempio (lo stesso di prima):

```
#include <stdio.h>
main()
    {
    char s[81];
    while (gets(s))  puts(s);
    }
```

- le **stringhe di caratteri** vengono memorizzate **in array di caratteri**
- le stringhe di caratteri **terminano con** il carattere **'\0'** (**NULL** - valore decimale zero)
- la gets **sostituisce** il **new line con** il **NULL**
- la puts **aggiunge un new line** alla stringa

```
putchar('A'); putchar('B'); puts("C"); putchar('D');
```

```
ABC
D
```

3. Input/Output con formato:

Si forniscono funzioni per la lettura/srittura di dati formattati di tipo molto diverso
si noti il numero variabile dei parametri

int **printf** (char *format, expr1, expr2, ..., exprN);

- scrive una serie di valori in base alle specifiche contenute in format
- i valori sono i risultati delle espressioni expr1, expr2, ..., exprN
- restituisce il numero di caratteri scritti, oppure EOF in caso di errore

int **scanf** (char *format, &var1, &var2, ..., &varN);

- legge una serie di valori in base alle specifiche contenute in format
- memorizza i valori nelle variabili var1, var2, ..., varN
  passate per riferimento
- restituisce il numero di valori letti e memorizzati, oppure EOF in caso di end-of-file

Esempi:

```
int k;
scanf("%d",&k);
printf("Il quadrato di %d e' %d",k,k*k);
```


Formati più comuni

		short	long
signed int	**%d**	**%hd**	**%ld**
unsigned int	**%u** (decimale)	**%hu**	**%lu**
	%o (ottale)	**%ho**	**%lo**
	%x (esadecimale)	**%hx**	**%lx**

float	**%e**, **%f**, **%g**
double	**%le**, **%lf**, **%lg**

carattere singolo	**%c**
stringa di caratteri	**%s**

puntatori (indirizzi) **%p**

Per l'output dei caratteri di controllo si usano:
'\n', '\t', etc.

Per l'output del carattere '%' si usa:
%% \% non funziona!

```
    printf("%x, %o, %%",70000,70000); /*  NO!  */
```
1, 10560, %

```
    printf("%lx, %lo, %%",70000,70000);      /*  SI!  */
```
11170, 210560, %

Esempi

```
main()
   {
    float x;
    int ret, i;
    char name[50];
   printf("Inserisci un numero decimale, ");
   printf("un floating ed una stringa con meno ");
   printf("di  50 caratteri e senza bianchi");
   ret = scanf("%d%f%s", &i, &x, name);
   printf("%d valori letti %d %f %s", ret, i, x, name);
   }
```

```
main()
   {
            int a;
   printf("Dai un carattere e ottieni il valore \
decimale, ottale e hex  ");
   a = getchar();
   printf("\n%c vale %d in decimale, %o in ottale \
e %x in hex.\n",a, a, a, a);
   }
```

Chi interpreta i caratteri % nelle stringhe di formato?

Accesso ad alto livello: strutture FILE

```
#include <stdio.h>
#define  MAX 80
main  (int argc; char ** argv)
{  char * f1, * f2;
    FILE * infile, * outfile;
     int nread, b;  char c, a, buf [MAX];
...
/* prologo: apertura dei file interessati */
/* le aree puntate da infile ed outfile non sono allocate  */
   if (( infile  = fopen ( f1, "r")) == NULL) exit (1);
   if (( outfile = fopen ( f2, "w")) == NULL)
      {  fclose (infile); exit (2); }

/*  operazioni a linea  */
   while ( fgets ( buf, MAX, infile) != NULL)
        fputs (buf, outfile); /* cioè uso di linee in I/O */
/* operazioni sicure, ma difficili da trattare */

/*  operazioni a blocchi */
   while ((nread = fread (buf, 1, MAX, infile)) > 0 )
         fwrite (buf, 1, MAX, outfile );

   while
    ((nitemread = fscanf (infile, "%c %d %c ", &a, &b, &c)) > 0 )
      if (nitemread == 3) fprintf (outfile, "%c %d %c ", a, b, c);

/* epilogo: chiusura dei file interessati */
  fclose (infile);  fclose (outfile);

}
```